INSTITUTO POLITÉCNICO NACIONAL

Unidad Profesional Interdisciplinaria de Ingeniería
Campus Zacatecas.

MATERIA:

- Ánalisis y Diseño de Algoritmos

Practica 03:

- Dijkstra

DOCENTE:

- Erika Sánchez Femat

Alumno:

- Gael García Torres

Introduccion

El algoritmo de Dijkstra es un método clásico en teoría de grafos utilizado para encontrar el camino más corto entre dos nodos en un grafo ponderado. Desarrollado por el científico Edsger Dijkstra en 1956, este algoritmo ha sido fundamental en numerosas aplicaciones, desde la planificación de redes de comunicación hasta la optimización de rutas en sistemas de transporte.

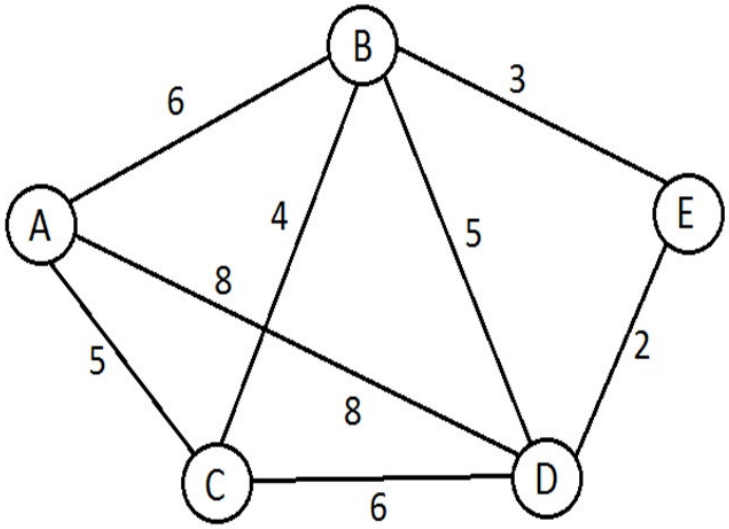
En esta práctica, nos sumergiremos en la implementación del algoritmo de Dijkstra utilizando el lenguaje de programación Python. El objetivo principal es comprender en detalle cómo funciona este algoritmo, desde la selección de nodos hasta la actualización de las distancias mínimas. Además, nos proponemos medir el tiempo de ejecución del algoritmo y analizar la complejidad computacional asociada.

Método de Dijkstra:

El algoritmo de Dijkstra, también llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de los vértices en un grafo con pesos en cada arista. Su nombre se refiere a Edsger Dijkstra, quien lo describió por primera vez en 1959.

La idea subyacente en este algoritmo consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices; cuando se obtiene el camino más corto desde el vértice origen, al resto de vértices que componen el grafo, el algoritmo se detiene. El algoritmo es una especialización de la búsqueda de costo uniforme, y como tal, no funciona en grafos con aristas de coste negativo (al elegir siempre el nodo con distancia menor, pueden quedar excluidos de la búsqueda nodos que en próximas iteraciones bajarían el costo general del camino al pasar por una arista con costo negativo).

Como funciona:



Para realizar la aplicación del algoritmo de Dijkstra, se aplican los siguientes pasos:

1. Se elige un nodo de inicio al cual se le marcara un peso de la siguiente forma:

$$[X,Y](N)$$

Donde 'X' equivale a el valor del recorrido actual de los arcos, 'Y' equivale a el nodo predecesor o de origen y 'N' al numero de iteración u operación actual

2. A los nodos adyacentes del nodo seleccionado como nodo de inicio, se deben asignar un peso de igual forma al punto anterior, ([X,Y](N))

3. De los nodos con los pesos calculados se toma el nodo con menor valor en X y este será el siguiente a visitar

4. Los pasos 2 y 3 deben repetirse teniendo en cuenta que si al intentar calcular los pesos para los nodos adyacentes a un nodo que esta siendo visitado, uno de estos ya tiene un peso asignado, deben calcularse los demás pesos cuantas veces sean necesario, y siempre se tomara el peso mínimo calculado

5. Los nodos pueden ser visitados una sola vez

Codigo:

```
import heapq
from collections import defaultdict
import time

class GrafoPonderado:
    def _init_(self):
        # Usamos un conjunto para los v rtices
        self.vertices = set()
        # Creamos un diccionario para las aristas con peso
        self.aristas = defaultdict(list)

    def agregar_vertice(self, vertice):
        self.vertices.add(vertice)

    def agregar_arista(self, desde_vertice, hacia_vertice, peso):
        self.vertices.add(desde_vertice)
        self.vertices.add(hacia_vertice)
        self.aristas[desde_vertice].append((hacia_vertice, peso))
        self.aristas[hacia_vertice].append((desde_vertice, peso))

    def dijkstra(self, vertice_inicial):
        distancias = {vertice: float('infinity') for vertice in self.vertices}
        distancias[vertice_inicial] = 0
        # Usamos defaultdict para los padres
        padres = defaultdict(lambda: None)
        cola_prioridad = [(0, vertice_inicial)]
        tiempo_inicio = time.time()

        while cola_prioridad:
            distancia_actual, vertice_actual = heapq.heappop(cola_prioridad)
            if distancia_actual > distancias[vertice_actual]:
                continue
            for vecino, peso_arista in self.aristas[vertice_actual]:
                distancia = distancia_actual + peso_arista
                if distancia < distancias[vecino]:
                    distancias[vecino] = distancia
                    padres[vecino] = vertice_actual
                    heapq.heappush(cola_prioridad, (distancia, vecino))

        tiempo_fin = time.time()
        tiempo_transcurrido = tiempo_fin - tiempo_inicio
        caminos_minimos = {vertice: self.construir_camino(vertice_inicial, vertice,

        padres) for vertice in self.vertices}
        return distancias, caminos_minimos, tiempo_transcurrido

    def construir_camino(self, vertice_inicial, vertice_final, padres):
        camino = []
        vertice_actual = vertice_final
        while vertice_actual is not None:
```

```
            camino.insert(0, vertice_actual)
            vertice_actual = padres[vertice_actual]
        return camino

# Ejemplo de uso del grafo
grafo = GrafoPonderado()
grafo.agregar_vertice("A")
grafo.agregar_vertice("B")
grafo.agregar_vertice("C")
grafo.agregar_vertice("D")
grafo.agregar_vertice("E")
grafo.agregar_vertice("F")
grafo.agregar_vertice("G")
grafo.agregar_vertice("H")
grafo.agregar_vertice("I")
grafo.agregar_arista("A", "B", 1)
grafo.agregar_arista("B", "C", 2)
grafo.agregar_arista("A", "C", 4)
grafo.agregar_arista("C", "D", 7)
grafo.agregar_arista("B", "E", 3)
grafo.agregar_arista("E", "F", 5)
grafo.agregar_arista("C", "G", 2)
grafo.agregar_arista("G", "H", 1)
grafo.agregar_arista("H", "I", 3)

vertice_inicial = "A"
distancias, caminos_minimos, tiempo_ejecucion = grafo.dijkstra(vertice_inicial)

print("**********\n")
print(f"El camino m s corto localizado es: {vertice_inicial}: {distancias}\n")
print("**********\n")

for vertice, camino in caminos_minimos.items():
    print("**********\n")
    print(f"El recorrido comienza desde {vertice_inicial} hasta {vertice}: {camino}\n")
print("**********\n")
print("**********\n")
print(f"El tiempo que tarda en ejecutarse es de: {tiempo_ejecucion} segundos")
print("**********\n")
```

Complejidad :

La complejidad temporal (tiempo de ejecución) del algoritmo de Dijkstra depende principalmente de la implementación y de cómo esté representado el grafo.

La complejidad temporal del algoritmo de Dijkstra se expresa comúnmente en términos de "O((V + E) * log(V))", donde "V" es el número de vértices y "E" es el número de aristas en el grafo.

- El bucle principal se ejecuta hasta que la cola de prioridad esté vacía, y en cada iteración, se extrae y actualiza un vértice y sus distancias a los vecinos. En la implementación estándar con un heap binario, cada extracción y actualización lleva tiempo "O(log(V))".

- El bucle externo se ejecutará tantas veces como aristas haya en el grafo, ya que cada arista se procesa una vez. Por lo tanto, el término "E" (número de aristas) multiplica la complejidad.

- La complejidad adicional de "log(V)" proviene de las operaciones en la cola de prioridad.

En resumen, la complejidad temporal del algoritmo de Dijkstra es eficiente y se puede manejar bien en la mayoría de los casos prácticos. Sin embargo, debes tener en cuenta que esta complejidad puede variar dependiendo de detalles específicos de implementación y del tipo de heap utilizado para la cola de prioridad.

Resultados:

```
************************************************************
El camino más corto localizado es: A: {'P': 51, 'Q': 61, 'A': 0, 'B': 15,
************************************************************
************************************************************
El recorrido comienza desde A hasta P: ['A', 'C', 'G', 'P']
************************************************************
El recorrido comienza desde A hasta Q: ['A', 'C', 'G', 'P', 'Q']
************************************************************
El recorrido comienza desde A hasta A: ['A']
************************************************************
El recorrido comienza desde A hasta B: ['A', 'B']
************************************************************
El recorrido comienza desde A hasta C: ['A', 'C']
************************************************************
El recorrido comienza desde A hasta D: ['A', 'C', 'D']
************************************************************
El recorrido comienza desde A hasta E: ['A', 'B', 'E']
************************************************************
El recorrido comienza desde A hasta F: ['A', 'B', 'E', 'F']
************************************************************
El recorrido comienza desde A hasta G: ['A', 'C', 'G']
************************************************************
El recorrido comienza desde A hasta H: ['A', 'C', 'G', 'H']
************************************************************
El recorrido comienza desde A hasta I: ['A', 'C', 'G', 'H', 'I']
************************************************************
```

```
************************************************************
El recorrido comienza desde A hasta I: ['A', 'C', 'G', 'H', 'I']
************************************************************
El recorrido comienza desde A hasta J: ['A', 'C', 'D', 'J']
************************************************************
El recorrido comienza desde A hasta K: ['A', 'C', 'D', 'J', 'K']
************************************************************
El recorrido comienza desde A hasta L: ['A', 'C', 'D', 'J', 'K', 'L']
************************************************************
El recorrido comienza desde A hasta M: ['A', 'B', 'E', 'M']
************************************************************
El recorrido comienza desde A hasta N: ['A', 'B', 'E', 'M', 'N']
************************************************************
El recorrido comienza desde A hasta O: ['A', 'B', 'E', 'M', 'N', 'O']
************************************************************
************************************************************
El tiempo que tarda en ejecutarse es de: 0.0 segundos
************************************************************
```

Resultado 1:

```
************************************************************
El camino más corto localizado es: A: {'P': 52, 'Q': 70, 'A': 0,
************************************************************
************************************************************
El recorrido comienza desde A hasta P: ['A', 'C', 'G', 'P']
************************************************************
El recorrido comienza desde A hasta Q: ['A', 'C', 'G', 'P', 'Q']
************************************************************
El recorrido comienza desde A hasta A: ['A']
************************************************************
El recorrido comienza desde A hasta B: ['A', 'B']
************************************************************
El recorrido comienza desde A hasta C: ['A', 'C']
************************************************************
El recorrido comienza desde A hasta D: ['A', 'C', 'D']
************************************************************
El recorrido comienza desde A hasta E: ['A', 'B', 'E']
************************************************************
El recorrido comienza desde A hasta F: ['A', 'B', 'E', 'F']
************************************************************
```

```
El recorrido comienza desde A hasta F: ['A', 'B', 'E', 'F']
************************************************************
El recorrido comienza desde A hasta G: ['A', 'C', 'G']
************************************************************
El recorrido comienza desde A hasta H: ['A', 'C', 'G', 'H']
************************************************************
El recorrido comienza desde A hasta I: ['A', 'C', 'G', 'H', 'I']
************************************************************
El recorrido comienza desde A hasta J: ['A', 'C', 'D', 'J']
************************************************************
El recorrido comienza desde A hasta K: ['A', 'C', 'D', 'J', 'K']
************************************************************
El recorrido comienza desde A hasta L: ['A', 'C', 'D', 'J', 'K', 'L']
************************************************************
El recorrido comienza desde A hasta M: ['A', 'B', 'E', 'M']
************************************************************
El recorrido comienza desde A hasta N: ['A', 'B', 'E', 'M', 'N']
************************************************************
El recorrido comienza desde A hasta O: ['A', 'B', 'E', 'M', 'N', 'O']
************************************************************
************************************************************
El tiempo que tarda en ejecutarse es de: 0.0 segundos
************************************************************
```

Resultado 2:

```
************************************************************
El camino más corto localizado es: A: {'P': 38, 'Q': 58, 'A': 0, 'B': 15,
************************************************************
************************************************************
El recorrido comienza desde A hasta P: ['A', 'C', 'G', 'P']
************************************************************
El recorrido comienza desde A hasta Q: ['A', 'C', 'G', 'P', 'Q']
************************************************************
El recorrido comienza desde A hasta A: ['A']
************************************************************
El recorrido comienza desde A hasta B: ['A', 'B']
************************************************************
El recorrido comienza desde A hasta C: ['A', 'C']
************************************************************
El recorrido comienza desde A hasta D: ['A', 'C', 'D']
************************************************************
El recorrido comienza desde A hasta E: ['A', 'B', 'E']
************************************************************
El recorrido comienza desde A hasta F: ['A', 'B', 'E', 'F']
************************************************************
El recorrido comienza desde A hasta G: ['A', 'C', 'G']
************************************************************
El recorrido comienza desde A hasta H: ['H']
************************************************************
El recorrido comienza desde A hasta I: ['I']
************************************************************
El recorrido comienza desde A hasta J: ['A', 'C', 'D', 'J']
************************************************************
```

```
El recorrido comienza desde A hasta J: ['A', 'C', 'D', 'J']
************************************************************
El recorrido comienza desde A hasta K: ['A', 'C', 'D', 'J', 'K']
************************************************************
El recorrido comienza desde A hasta L: ['A', 'C', 'D', 'J', 'K', 'L']
************************************************************
El recorrido comienza desde A hasta M: ['A', 'B', 'E', 'M']
************************************************************
El recorrido comienza desde A hasta N: ['A', 'B', 'E', 'M', 'N']
************************************************************
El recorrido comienza desde A hasta O: ['A', 'B', 'E', 'M', 'N', 'O']
************************************************************
************************************************************
El tiempo que tarda en ejecutarse es de: 0.0 segundos
************************************************************
```

Resultado 3:

```
************************************************************
El camino más corto localizado es: A: {'P': 49, 'Q': 63, 'A': 0, 'B':
************************************************************
************************************************************
El recorrido comienza desde A hasta P: ['A', 'C', 'G', 'P']
************************************************************
El recorrido comienza desde A hasta Q: ['A', 'C', 'G', 'P', 'Q']
************************************************************
El recorrido comienza desde A hasta A: ['A']
************************************************************
El recorrido comienza desde A hasta B: ['A', 'B']
************************************************************
El recorrido comienza desde A hasta C: ['A', 'C']
************************************************************
El recorrido comienza desde A hasta D: ['D']
************************************************************
El recorrido comienza desde A hasta E: ['A', 'B', 'E']
************************************************************
El recorrido comienza desde A hasta F: ['A', 'B', 'E', 'F']
************************************************************
El recorrido comienza desde A hasta G: ['A', 'C', 'G']
************************************************************
El recorrido comienza desde A hasta H: ['A', 'C', 'G', 'H']
************************************************************
El recorrido comienza desde A hasta I: ['A', 'C', 'G', 'H', 'I']
************************************************************
```

```
El recorrido comienza desde A hasta I: ['A', 'C', 'G', 'H', 'I']
************************************************************
El recorrido comienza desde A hasta J: ['J']
************************************************************
El recorrido comienza desde A hasta K: ['K']
************************************************************
El recorrido comienza desde A hasta L: ['L']
************************************************************
El recorrido comienza desde A hasta M: ['A', 'B', 'E', 'M']
************************************************************
El recorrido comienza desde A hasta N: ['A', 'B', 'E', 'M', 'N']
************************************************************
El recorrido comienza desde A hasta O: ['A', 'B', 'E', 'M', 'N', 'O']
************************************************************
************************************************************
El tiempo que tarda en ejecutarse es de: 0.0 segundos
************************************************************
```

Resultado 4 :

```
************************************************************
El camino más corto localizado es: A: {'P': 53, 'Q': 65, 'A': 0, 'B': 10,
************************************************************
************************************************************
El recorrido comienza desde A hasta P: ['A', 'C', 'G', 'P']
************************************************************
El recorrido comienza desde A hasta Q: ['A', 'C', 'G', 'P', 'Q']
************************************************************
El recorrido comienza desde A hasta A: ['A']
************************************************************
El recorrido comienza desde A hasta B: ['A', 'B']
************************************************************
El recorrido comienza desde A hasta C: ['A', 'C']
************************************************************
El recorrido comienza desde A hasta D: ['A', 'C', 'D']
************************************************************
El recorrido comienza desde A hasta E: ['A', 'B', 'E']
************************************************************
El recorrido comienza desde A hasta F: ['A', 'B', 'E', 'F']
************************************************************
El recorrido comienza desde A hasta G: ['A', 'C', 'G']
************************************************************
El recorrido comienza desde A hasta H: ['A', 'C', 'G', 'H']
************************************************************
El recorrido comienza desde A hasta I: ['A', 'C', 'G', 'H', 'I']
************************************************************
```

```
El recorrido comienza desde A hasta I: ['A', 'C', 'G', 'H', 'I']
************************************************************
El recorrido comienza desde A hasta J: ['A', 'C', 'D', 'J']
************************************************************
El recorrido comienza desde A hasta K: ['A', 'C', 'D', 'J', 'K']
************************************************************
El recorrido comienza desde A hasta L: ['A', 'C', 'D', 'J', 'K', 'L']
************************************************************
El recorrido comienza desde A hasta M: ['A', 'B', 'E', 'M']
************************************************************
El recorrido comienza desde A hasta N: ['A', 'B', 'E', 'M', 'N']
************************************************************
El recorrido comienza desde A hasta O: ['A', 'B', 'E', 'M', 'N', 'O']
************************************************************
************************************************************
El tiempo que tarda en ejecutarse es de: 0.0 segundos
************************************************************
```

Conclusion :

En esta práctica, hemos llevado a cabo la implementación del algoritmo de Dijkstra para la búsqueda de caminos más cortos en un grafo ponderado. Este ejercicio ha sido fundamental para consolidar nuestra comprensión sobre el funcionamiento del algoritmo y su aplicación práctica utilizando el lenguaje de programación Python.

A lo largo del proceso, se realice mejoras en el código original, corrigiendo errores tipográficos y reorganizando la estructura para aumentar la claridad y eficiencia. También amplie el grafo, introduciendo más datos, con el propósito de realizar pruebas más exactas.